7. Protokoly aplikační vrstvy

**1. Účel, funkce a charakteristika protokolů aplikační vrstvy**

**Účel:**

umožňuje aplikacím komunikaci a přenos dat mezi sebou bez ohledu na fyzickou síťovou infrastrukturu

* **Standardizace:** Protokoly aplikační vrstvy definují standardy a formáty, které umožňují různým softwarovým aplikacím komunikovat bez ohledu na jejich vnitřní implementaci.
* **Přenos a prezentace dat:** Umožňují transformaci dat z interní reprezentace na standardizovaný formát (např. JSON, XML, HTML), který je srozumitelný oběma komunikujícími stranami.
* **Zajištění specifických služeb:** Každý aplikační protokol je navržen s ohledem na konkrétní typ služby – ať už jde o webové stránky, přenos souborů, e-mailovou komunikaci, nebo ovládání vzdálených zařízení.

**Funkce:**

* **Formátování a kódování:** Protokoly definují způsob, jakým se data převádějí do přenositelné podoby. Například HTTP používá textový protokol s hlavičkami a tělem zprávy, zatímco SOAP využívá XML pro strukturovanou výměnu informací.
* **Řízení relací a stavů:** Některé protokoly, například FTP nebo SMTP, implementují mechanismy pro udržování a řízení relací – od navázání spojení, přes autentizaci, až po ukončení komunikace.
* **Řízení chyb a potvrzování:** I když základní chyby a retransmise řeší transportní vrstva (TCP/UDP), aplikační protokoly často implementují vlastní mechanismy pro zpracování specifických chyb (např. kódy stavu HTTP – 404, 500, atd.) a pro ověření integrity či konzistence dat.
* **Bezpečnostní mechanismy:** Některé protokoly zahrnují integrované bezpečnostní prvky, jako je šifrování (HTTPS, FTPS), ověřování uživatele (SMTP autentizace) nebo využití tokenů a certifikátů pro ověření identity.

**Charakteristika:**

protokoly aplikační vrstvy jsou navrženy tak, aby byly nezávislé na konkrétních technologiích nebo protokolech nižších vrstev

* **Specifičnost k dané službě:** Protokoly jsou optimalizovány pro konkrétní druh komunikace – například DNS je navržený pro překlad doménových jmen, zatímco HTTP je určeno pro přenos hypertextových dokumentů.
* **Standardizace a rozšířenost:** Většina protokolů aplikační vrstvy je definována v dokumentech RFC, což umožňuje širokou implementaci napříč různými platformami a zařízeními.

**Klient-server architektura**

* **Princip:**
  + Komunikace probíhá mezi centralizovaným serverem, který poskytuje služby, a klienty, kteří tyto služby využívají.
  + Servery mají speciální úlohu a jsou optimalizovány pro správu dat, autentizaci a zabezpečení.
* **Příklady:**
  + Webové servery (HTTP/HTTPS), e-mailové servery (SMTP, IMAP), databázové servery.
* **Výhody:**
  + Centralizovaná správa, bezpečnost a kontrola nad přístupem.
  + Snadnější implementace záloh, monitorování a údržby.
* **Nevýhody:**
  + Centralizovaný bod selhání, vyžaduje robustní serverovou infrastrukturu.

**Peer-to-peer (P2P) architektura**

* **Princip:**
  + V P2P síti se každé zařízení (peer) chová zároveň jako klient i server.
  + Data jsou sdílena přímo mezi zařízeními bez centralizovaného serveru.
* **Příklady:**
  + Sdílení souborů (např. BitTorrent), některé decentralizované aplikace.
* **Výhody:**
  + Decentralizovaná správa, vyšší odolnost vůči selhání jednoho bodu.
  + Možnost efektivního rozložení zatížení mezi všechny uzly.
* **Nevýhody:**
  + Obtížnější správa bezpečnosti a autentizace, potenciálně nižší kontrola nad kvalitou a dostupností dat.

**2. Přehled protokolů aplikační vrstvy**

**HTTP/HTTPS (HyperText Transfer Protocol / Secure)**

* **Funkce:**
  + Slouží k přenosu webových dokumentů mezi klientem (prohlížečem) a serverem.
  + HTTPS využívá TLS/SSL pro šifrování komunikace, což zvyšuje bezpečnost přenosu citlivých informací.
* používá port TCP/**80**
* **URL** – jednotný lokátor prostředků, specifikuje jednoznačné umístění zdroje v internetu
* **Princip činnosti**
  + klient (webový prohlížeč) posílá žádost o určitý zdroj (webová stránka) na server
  + server odpovídá odpovědí obsahující požadovaný zdroj
* metody HTTP požadavků - GET, POST, PUT, DELETE…
* HTTP je protokol **bezestavový** - server si neukládá žádné informace o předchozích požadavcích a odpovědích, které byly vyměněny mezi klientem a serverem
* **HTTPS**
  + **zabezpečená verze HTTP** protokolu
  + spojení mezi klientem a serverem je **šifrované**
  + data se přenáší v zašifrované formě pomocí protokolu **SSL/TLS**

**FTP, FTPS, SFTP (File Transfer Protocol)**

* **FTP:**
  + Původní protokol pro přenos souborů, který využívá samostatné kanály pro příkazy a data (port TCP/**21 (spojení)** a TCP/**20 (data)**).
* **FTPS:**
  + Rozšíření FTP s využitím **TLS/SSL** pro bezpečný přenos dat.
* **SFTP:**
  + Protokol pro bezpečný přenos souborů založený na **SSH**, poskytující šifrovanou komunikaci a autentizaci.

**TFTP (Trivial File Transfer Protocol)**

* **TFTP** je zjednodušený protokol pro přenos souborů, určený pro situace, kde není třeba robustního řízení spojení či autentizace.
  + **Charakteristika:**
    - **Používá UDP:** TFTP pracuje na bázi UDP (obvykle port 69)
    - **Použití:** Často se využívá při síťovém bootování (PXE boot) a při aktualizaci firmware zařízení, kdy stažení malého souboru bez složitých ověřovacích mechanismů postačuje.
  + **Výhody a nevýhody:**
    - **Výhody:** Nízká režie, jednoduchá konfigurace.
    - **Nevýhody:** Chybí bezpečnostní funkce, ověřování a mechanismy řízení chyb, což jej činí méně vhodným pro přenos citlivých dat.

**E-mailové protokoly: SMTP, POP3, IMAP**

* **SMTP (Simple Mail Transfer Protocol):**
  + Zajišťuje **odesílání e-mailů mezi servery**. (**port 25**)
  + Podporuje rozšíření, jako je SMTP AUTH pro autentizaci.
* **POP3 (Post Office Protocol 3):**
  + Umožňuje **stahování e-mailů** **ze serveru do klientského zařízení**.
  + Obvykle stahuje a poté odstraní zprávy ze serveru.
* **IMAP (Internet Message Access Protocol):**
  + Poskytuje komplexnější správu e-mailů přímo na serveru, umožňuje synchronizaci mezi více zařízeními.

**DNS (Domain Name System)**

* **Funkce:**
  + Překládá doménová jména na IP adresy a obráceně. (port 53)
  + **funguje pomocí hierarchické struktury**
  + Distribuovaný systém, který zajišťuje rychlý a efektivní dotazovací proces.
* **Bezpečnost:**
  + DNSSEC přidává digitální podpisy k ochraně před manipulací s daty.
* Lze pro něj použít jak TCP, tak i UDP
  + TCP: např. když potřebujeme přenést záznamy zonových souborů na jiný server
  + UDP: komunikace klient/server, nepotřebujeme navazovat spojení, a tak vyžaduje menší režii

**Telnet a SSH (Secure Shell)**

* **Telnet:**
  + Původní protokol pro vzdálený přístup, který však neposkytuje šifrování.
* **SSH:**
  + Bezpečná náhrada za Telnet, která nabízí šifrovanou komunikaci, autentizaci a další bezpečnostní funkce.

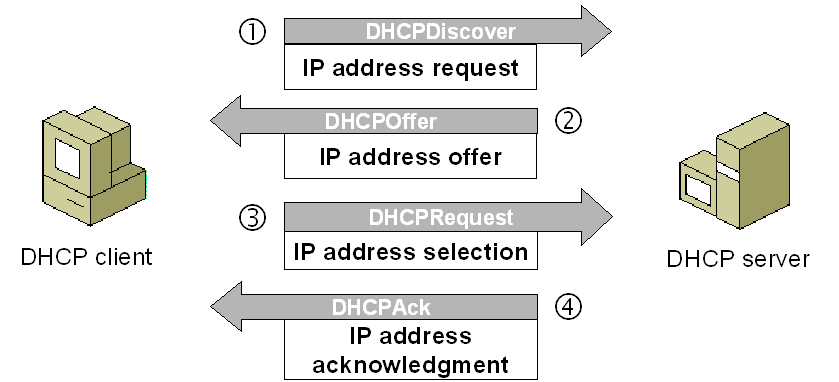
**DHCP**

* Dynamic Host Configuration Protocol
* síťový protokol **pro automatickou konfiguraci zařízení v síti** (počítače, mobilní telefony, tiskárny…)
  + **přiřazuje IP adresu, masku podsítě, adresu výchozí brány a adresu DNS serveru**
* umožňuje správci sítě efektivně spravovat adresní prostor a zabránit kolizím IP adres

**Princip činnosti**

* DHCP klienti žádají DHCP server o IP adresu
* DHCP server odpoví přidělením konfigurace a zaeviduje dobu zapůjčení IP adresy
* po vypršení doby může adresu přiřadit jinému zařízení

1. po připojení do sítě klient vyšle broadcastem DHCPDISCOVER paket
2. DHCP server odpoví paketem DHCPOFFER s nabídkou IP adresy
3. klient si vybere IP adresu a o tu požádá paketem DHCPREQUEST
4. server mu ji potvrdí odpovědí DHCPACK
5. jakmile klient obdrží DHCPACK, může už IP adresu a zbylá nastavení používat



**DHCP Relay Agent**

* síťový prvek, který umožňuje přenos DHCP požadavků a odpovědí mezi DHCP klientem a serveru na jiných sítích
* protože routery neumožňují broadcast mezi sítěmi, je možnost použít DHCP Relay Agent
* může být implementován jako samostatný síťový prvek nebo může být součástí routeru

**SNMP (Simple Network Management Protocol)**

* **Funkce:**
  + Umožňuje správu a monitorování síťových zařízení.
  + Používá se ke sběru dat o stavu zařízení, jako jsou routery, switche, servery a další.
* **Verze:**
  + SNMPv1 a SNMPv2 poskytují základní funkce, zatímco SNMPv3 přináší pokročilé bezpečnostní mechanismy.

**3. Výběr transportního protokolu pro aplikace**

Výběr vhodného transportního protokolu je klíčový pro zajištění správné funkčnosti a efektivity komunikace aplikací. Klíčová kritéria zahrnují spolehlivost, rychlost, režii a bezpečnost.

**TCP (Transmission Control Protocol)**

* **Charakteristika:**
  + Zajišťuje spolehlivý, orientovaný na spojení přenos dat.
  + Implementuje mechanismy pro kontrolu chyb, sekvenční doručování paketů a řízení toku (flow control).
  + Využívá potvrzovací zprávy (ACK) a retransmise, pokud dojde ke ztrátě paketů.
* **Použití:**
  + Vhodný pro aplikace, kde je kritická integrita dat – webové stránky, e-mail, FTP, databázová připojení.
* **Režie:**
  + Vyšší režie kvůli mechanismům řízení spojení, což může být nevýhodné pro aplikace vyžadující minimální latenci.

**UDP (User Datagram Protocol)**

* **Charakteristika:**
  + Poskytuje nespolehlivý, bezspojový přenos dat, bez záruky doručení nebo pořadí paketů.
  + Má minimální režii, což zajišťuje rychlejší přenos dat.
* **Použití:**
  + Vhodný pro aplikace, kde rychlost přenosu převyšuje potřebu spolehlivosti – například streamování videa, VoIP, online hry.
* **Výhody a nevýhody:**
  + Na jedné straně UDP umožňuje velmi nízkou latenci a vyšší rychlost, ale na druhé straně nedokáže automaticky řešit ztrátu dat.

**Kritéria výběru:**

* **Požadovaná spolehlivost:** Pokud aplikace nemůže tolerovat ztrátu dat (např. finanční transakce nebo důležité obchodní zprávy), je vhodnější TCP.
* **Latence a rychlost:** Aplikace jako videokonference nebo online hry, kde je klíčová minimální latence, mohou využít UDP, i když se mohou vyskytnout drobné ztráty dat.
* **Řízení provozu a škálovatelnost:** V sítích s vysokou zátěží může být nutné použít protokoly, které efektivně řídí přenos a reagují na změny v síťových podmínkách (např. congestion control mechanismy v TCP).
* **Zabezpečení:** využití SSL/TLS

**UDP**

* DNS, DHCP, SNMP, TFTP

**TCP**

* DNS, HTTP (tady pozor, nové verze HTTP se zakládají na UDP), SMTP, FTP, SSH, Telnet

**Shrnutí**

* **Protokoly aplikační vrstvy** slouží jako most mezi koncovými aplikacemi a základními síťovými službami. Zajišťují formátování, řízení relací, autentizaci a správu chyb, čímž umožňují aplikacím komunikovat bez ohledu na vnitřní implementaci.
* **Přehled protokolů** zahrnuje širokou škálu služeb: HTTP/HTTPS pro web, FTP/SFTP/FTPS pro přenos souborů, e-mailové protokoly (SMTP, POP3, IMAP), DNS pro překlad jmen, SSH pro bezpečný vzdálený přístup a aplikační protokoly jako REST a SOAP.
* **Výběr transportního protokolu** je zásadní – TCP je preferováno pro spolehlivé přenosy dat, zatímco UDP je vhodnější tam, kde je prioritou rychlost a nízká latence.